How Reliable is Smartphone-based Electronic Contact Tracing for COVID-19?

A Look through the Lens of Neighbor Discovery Protocols

Philipp H. Kindt
Chemnitz University of Technology, Germany
philipp.kindt@informatik.tu-chemnitz.de

Trinad Chakraborty
Justus-Liebig University Giessen, & German Center of Infection Research
trinad.chakraborty@mikrobio.med.uni-giessen.de

Samarjit Chakraborty
University of North Carolina at Chapel Hill, USA
samarjit@cs.unc.edu

ABSTRACT

Smartphone-based contact tracing is currently considered an essential tool towards easing lockdowns, curfews, and shelter-in-place orders issued in response to the 2020/21 novel coronavirus (SARS-CoV-2) crisis. While the focus on developing smart-phone-based contact tracing applications has been on privacy concerns, an important question that has not received sufficient attention is: *How reliable is such smartphone-based electronic contact tracing?* Here, we examine in detail the technical prerequisites required for effective smartphone-based contact tracing. The underlying mechanism that any tracing app relies on is called *neighbor discovery* (ND), which involves smartphones transmitting and scanning for Bluetooth signals to record their mutual presence. However, smartphones were not designed for reliable contact tracing. In this paper, we quantitatively evaluate how reliably smartphones can do contact tracing. We show that irrespective of how well a contact tracing app is designed, because of the limitations stemming from how ND mechanisms are implemented on currently-available phones, they cannot achieve *gapless* tracing. We also discuss the design of a wearable solution, which could lead to a more reliable tracing.

1 INTRODUCTION

Background: The global surges of the novel coronavirus SARS-CoV-2 in 2020 and 2021 have led to a partial, and at some places even a complete lockdown across the world. Since every infected person can potentially cause multiple secondary infections, the solution adopted is to limit social contacts by enforcing social distancing and stay-at-home regimes. A promising solution being considered to enable the gradual easing or prevention of future lockdowns is wireless contact tracing using smartphones. If all relevant previous contacts of a person tested positive for the virus are quickly and reliably identified and isolated, then any further spread of the infection will be reduced. Indeed, studies have shown that smartphone-based contact tracing can help to control the spread of infectious diseases [25].

When modeling the spread of infections in a pandemic, every infected person is considered to infect R others, where R is referred to as the *effective reproduction rate*. Recent studies, e.g., [6], suggest that the value of R can be reduced using electronic contact tracing. The extent of this reduction depends on two factors. The first is the fraction of the population that uses a smartphone-based contact tracing application or app, i.e., its adoption rate. The second factor is determined by the probability with which a smartphone is able to reliably detect a contact, which again depends on multiple factors, which we discuss below. While the current discussion has focused on the adoption rates, most work on modelling the effect of electronic contact tracing has implicitly assumed that whenever a smartphone is used, it *reliably* registers all or at least a certain fraction of relevant contacts. The validity of this reliability assumption has however not been sufficiently studied so far, and is the focus of this paper.

Reliability of Contact Tracing: There are multiple factors that influence the reliability of contact tracing. First, as mentioned above, a sufficiently high fraction of the population needs to install the app. Second, there is a minimum duration for which two devices need to be within range of each other for being able to detect their mutual presence. This paper addresses this aspect in detail by quantifying such durations and by studying secondary aspects, such as the corresponding battery runtime. Third, when multiple smartphones are within the range of reception, their emitted signals interfere with each other, thereby potentially preventing their mutual detection. We also study the impact of collisions on the discovery procedure. Fourth, after two devices have discovered each other, they estimate their distance to evaluate the transmission risk. The accuracy of this estimation also contributes to the reliability, which is also discussed in this paper.

Neighbor Discovery – The Contact Tracing Basis: Because localization systems, such as the global positioning system (GPS), are inaccurate inside buildings, devices in vicinity are detected using short-range wireless signals. The mechanism that lies at the heart of a smartphone detecting the existence of another smartphone in its vicinity is called *neighbor discovery (ND)*. It is based on the phones emitting and scanning for Bluetooth signals, and a successful reception by another phone and vice-versa will lead to their mutual discovery. Both the specific ND protocol and the manner in which it is parameterized determine the discovery latency, i.e., how quickly two smartphones will be able to "discover" each other when they come in close contact, and also their energy consumption and hence battery runtime. Other properties, such as the reliability of operation when a large number of phones are in range of reception, are also determined by this. However, smartphones constrain the degrees of freedom when realizing a protocol for ND, such that these aspects cannot be optimized by an app.

Limitations of Smartphones: Hence, when a smartphone is used for contact tracing, a relevant question is: whenever two or more people come in contact with each other, what is the probability of their respective smartphones being able to record such a contact?

If the duration of the contact is very brief (e.g., a couple of seconds), would such contacts still be detected? Will smartphones be able to detect the *type* and proximity of the contact? For example, were the two subjects within 5 meters of each other or at less than 0.1 meters? The answers to these questions in the context of smartphone-based contact tracing are not clear. Nevertheless, contact tracing apps are now being seen as the holy grail for this problem. However, what a tracing app might or might not be able to do will be fundamentally limited by the underlying hardware and protocol configurations of ND mechanisms supported by the smartphone.

Besides the success probability of the ND procedure itself, the estimation of the distance between two subjects is susceptible to errors. This estimation is based on sensing the attenuation of the wireless signal, which in free space correlates with the distance between two devices. However, signals can be reflected in the environment or attenuated by the human body. Hence, contacts that are not relevant could be misclassified as relevant, (viz., result in *false positives*), and actually relevant contacts might not be registered. However, too many false positives would lead to testing and/or isolating a large number of uninfected people. Hence, a sufficient "safety margin" needs to be built into the distance estimation procedure, which in turn will lead to a higher rate of unregistered relevant contacts.

Contributions of this Paper: In this paper, we attempt to address the above questions. We systematically evaluate the suitability of ND configurations in commercially-available smartphones for the purpose of electronic contact tracing. Our study exposes the fundamental limits that any smartphone will have, no matter which contact tracing app is used. We finally argue that for "gapless" contact tracing, smartphones are not suitable. Though the performance of the ND procedure on smartphones could improve if Google and Apple would grant unrestricted access to the phone's Bluetooth capabilities, distance estimation will nevertheless remain unreliable. Driven by this insight that *detection reliability* is a critical variable, we discuss a wearable solution for contact tracing, such as an electronic bracelet, which can overcome the limitations of smartphones.

Summary and Organization: In summary, our main contributions are as follows. (a) We debunk the currently held notion that smartphones can reliably conduct contact tracing and the only obstacle is their adoption. (b) Towards this, we lay the foundations for quantifying the reliability and accuracy of contact tracing when using currently-available smartphones. (c) We briefly discuss a dedicated wearable for contact tracing, which could resolve some of the limitations of smartphones.

The rest of this paper is organized as follows. In the next section, we briefly describe the theoretical foundations of energy- and latency-optimal solutions for contact tracing. In Section 3, the performance of contact tracing using currently-available `iOS` and `Android` smartphones is evaluated. In Section 4, we propose a wearable device for contact tracing before concluding in Section 5.

2 NEIGHBOR DISCOVERY

The goal of this section is to illustrate the design space of the ND procedure, which underlies electronic contract tracing. It involves multiple trade-offs, e.g., latency versus energy consumption versus resilience in crowded situations. Any smartphone application for contact tracing will build on a restricted version of this procedure, which we study in Section 3.

Let us first consider two wireless devices that are unaware of their mutual presence, but would like to "discover" each other as soon as they are in close proximity. One of them acts as a *sender* and the other as a *receiver*. The sender continually broadcasts beacons, while the receiver continually listens to the channel for certain time intervals. All transmissions and receptions are scheduled following a certain, repetitive pattern. The receiver has discovered the sender, once a beacon is sent within a reception window of the receiver.

The main reason behind both transmission and listening being *continual* and not *continuous* is energy. When not sending or receiving, the wireless radios go to a *sleep* or power-down mode in order to save energy. From the perspective of *ND*, the energy consumption of each device is determined by the fraction of time spent on transmission or reception, i.e., by the *duty-cycle* for transmission β and for reception γ. For a *ND* protocol to be *efficient*, the goal is to identify a transmission and reception pattern that, for a given sum $\beta + \gamma$, minimizes the time until a beacon is guaranteed to overlap with a reception window in the worst case. Several publications have been concerned with optimizing this trade-off between latency and energy, e.g., [4, 10]. In the context of contact tracing, the ND procedure should guarantee discovery within a short period of time. In other words, it should be able to register contacts even when two people come in close proximity for relatively short intervals of time, e.g., when shopping at a grocery store, or jogging in a park. We next discuss the latency within which a smartphone should guarantee the discovery of a remote device.

Required Latency for Contact Tracing: There is no established consensus on the minimum duration of a contact that results in significant transmission risk. For example, the American Center for Disease Control and Prevention (CDC) considers close contacts of a cumulative duration of at least 15 min within 24 h as relevant for transmitting SARS-CoV2. However, data from several studies show that there is a transmission risk also for brief contacts. Results obtained using manual contact tracing [3] indicate 5 out of 199 transmissions with a contact duration below 15 min. Moreover, there are reports [15] claiming that a fleeting contact of only a few seconds has lead to an infection with the contagious Delta variant. Hence, there is no clear threshold below which a contact is not relevant for transmission.

Since the transmission risk increases with the accumulated contact duration, i.e., the sum of the durations of multiple brief contacts, a system for contact tracing should ideally be able to record *every* encounter between two subjects and then estimate the risk based on the recorded data. Otherwise, multiple brief contacts, which can accumulate to a significant duration, risk getting unnoticed. When two subjects pass each other with zero distance at the closest point, with a velocity of e.g., 4 km/h, and if the range for tracing relevant contacts would be 2 m, they would stay within transmission range for 1.8 s. Hence, as a rule of thumb, contact tracing systems need to guarantee worst-case latencies of around 2 s for reliably detecting *all* contacts. While larger latencies in the range of up to tens of seconds might still be sufficient in many practical scenarios, minute-range latencies are not sufficient for *gapless* tracing.

Latency-Energy Trade-off: However, being able to do so should not quickly drain the battery of the device. What is the lowest discovery latency L that any receiver with a duty-cycle of γ and any sender with a duty-cycle of β can achieve was determined in [11] and is as follows.

$$L = \left\lceil \frac{1}{\gamma} \right\rceil \cdot \frac{\omega}{\beta} + \omega, \quad (1)$$

where ω is the transmission duration of a beacon. For example, if we require that only contacts that last for at least 2 seconds are relevant for disease transmission and need to be registered, then $L = 2$ s. The minimal data to be exchanged by two contact tracing devices in range is a device identifier. To be able to provide at least one unique ID for each device, 4 bytes or more are required. In addition, a preamble of at least 1 byte needs to be added for technical reasons on most radios. Hence, we assume that a 5 byte-packet needs to be transmitted for contact tracing on a Bluetooth Low Energy (BLE)-like physical layer, leading to a transmission duration of $\omega = 40$ µs. In order to realize $L = 2$ s with a beacon length of $\omega = 40$ µs , both devices need to be active for $\beta = \gamma \approx 0.45$ % of their time.

Using which transmission and reception patterns can this performance be achieved? The only known patterns that achieve optimal discovery latencies are based on *periodic intervals* [10]. Here, one device periodically broadcasts beacons with a period T_a, whereas the other device switches on its receiver for a window of d_s time-units after every T_s time-units. This scheme – with some minor modifications that we describe below – is used by the BLE protocol implemented inside smartphones.

Recall that for one sender and one receiver, the optimal discovery latency for a given energy-budget is known (cf. Equation 1), and this latency is guaranteed in 100 % of all discovery attempts. But the moment both devices act as *both* senders and receivers, the probability of discovery within the same time interval L now drops. Why is this so is explained next.

First, when each device both receives and transmits, it is unavoidable that the scheduled points in time of a reception window and a beacon transmission of the same device overlap in time (cf. [11] for details). Since a device cannot receive and transmit at the same time, some transmission or reception windows need to be re-scheduled, which leads to a certain fraction of discoveries failing. Nevertheless, existing protocols can achieve latencies close to L from Equation 1 in more than 99.9 % of all attempts for a pair of devices discovering each other using practical duty-cycles. However, when multiple devices are in range and transmit beacons, some transmissions might overlap in time and hence fail to be received. This is described next.

2.1 ND in Crowded Scenarios

There are multiple situations that are of potential relevance for virus transmission, in which a larger number of people are in vicinity of each other. For example, consider a crowed public bus or ski gondola. As a worst-case scenario, the maximum density of crowds without squashing and tilting the human body has been estimated to be 6 persons/m^2 [20]. If we assume that the radio only detects devices within a 2 m range, the worst-case number of people/phones in a collision domain is 75.

For a given number of devices, the probability of collisions is determined by the transmission duration of each beacon ω and the rate of beacons transmissions of each device, which result in a certain duty-cycle for transmission β. With $\omega = 40$ µs, when using known approaches for realizing $L = 2$ s, e.g., [10], about 50 % of all discovery attempts will fail due to colliding beacons (cf. [11]). This implies that a significant number of contacts will not be registered. However, if some increase in the worst-case latency L is tolerated, the following two techniques, which are used in smartphones, can be exploited for making ND protocols more robust against collisions.

Reducing the Channel Utilization: The probability of collisions is given by the fraction of time each device utilizes the channel, i.e., β. Hence, if fewer beacons are sent, then the collision probability decreases. As a drawback, since beacons are then sent less frequently, the worst-case latency L will increase, or the duty-cycle for reception γ needs to be increased for compensating for the reduced β (see Equation 1). Reducing β and in turn increasing γ for reaching the same L will, however, increase the overall energy consumption. For example, when choosing a reduced channel-utilization of $\beta' = 1/4 \cdot \beta$ and an increased duty-cycle for reception of $\gamma' = 4 \cdot \gamma$, the collision probability for the protocol described above when simultaneously discovering 75 devices can be reduced from 50 % to about 15 % without increasing L. On a radio on which transmission consumes the same current as reception, the lowest worst-case latency for a given energy budget $\eta = \beta + \gamma$ is achieved for $\beta = \gamma = 1/2(\beta + \gamma)$ [11]. Reducing the channel utilization to β' while increasing γ to γ' will hence lead to an increased sum $\eta' = \beta' + \gamma' = 17/8 \cdot \eta$ and therefore to an increased energy consumption, while leading to the same L. As we will show in Section 3, smartphones use a very low channel utilization $\beta << (\beta+\gamma)/2$, thereby achieving low collision rates at the expense of an increased energy consumption.

Decorrelation: A technique used by the BLE protocol is decorrelation. It reduces the chance of multiple subsequent colliding beacons. Instead of sending beacons with periodic intervals, two consecutive beacons are separated from each other by a fixed amount of time plus a certain random component. For nevertheless guaranteeing the same worst-case latency L, the reception duty-cycle γ needs to be increased. Consider, for example, a configuration where $T_a = d_s - \omega$, which has been shown to be a configuration that achieves the smallest possible worst-case latency [11]. Here, a beacon will coincide with *every* reception window, since the distance between two consecutive beacons does not exceed the reception window length. If now T_a is additionally increased by a random component $\rho \in [0, b]$, then d_s also needs to be extended by b time-units to ensure that a beacon will still fall within every scan window. Otherwise, another beacon has to overlap with a later scan window for realizing discovery, thereby increasing L. However, if one pair of beacons from two different devices collides, a later pair of beacons only collides with a probability below unity. If now multiple beacons coincide with one or more scan windows of a remote device, there is an increased chance that one of them will not collide and hence, the probability of a successful discovery increases.

In summary, there are multiple degrees of freedom for optimizing the ND procedure, and identifying the optimal trade-off between discovery latency, energy consumption and success probability is crucial for efficient and reliable contact tracing. Unfortunately, on a smartphone, these degrees of freedom are not exposed to a

Mode	$T_{a,0}$	d_s	T_s
ADVERTISE_MODE_LOW_LATENCY	100	-	-
ADVERTISE_MODE_BALANCED	250	-	-
ADVERTISE_MODE_LOW_POWER	1000	-	-
SCAN_MODE_LOW_POWER	-	512	5120
SCAN_MODE_BALANCED	-	1024	4096
SCAN_MODE_LOW_LATENCY	-	4096	4096
GAEN (mean)	279	3587	259694

Table 1: Parameter settings supported by Android and GAEN in [ms].

contact tracing app and several protocol parameters are already predetermined. In light of this, we examine the performance that is achievable using smartphones in the next section.

3 DISCOVERY ON SMARTPHONES

On a smartphone, the degrees of freedom described in the previous section are restricted both by the BLE protocol, which is used to realize ND, as well as the Android or iOS operating system. We first describe these restrictions and then study the performance achieved under them. Our goals are i) assessing the tracing performance on existing smartphones, and ii) identifying the most suitable parameterizations.

3.1 Restrictions on Smartphones

3.1.1 Bluetooth Low Energy. In BLE, so-called *advertising events*, within which beacons are sent, are scheduled with a period of T_a [16]. Reception windows of length d_s are repeated with a period of T_s. Here, T_a is composed of a static part $T_{a,0}$ plus a random delay of $\rho \in [0, 10\,\text{ms}]$. Furthermore, each advertising event consists of a sequence of three beacons. Each of them is sent on a dedicated wireless channel, and these three channels are the same for all devices and advertising events. The receiver toggles between the three different channels after each instance of T_s. The values of $T_{a,0}$, T_s and d_s can be chosen freely within a large, quasi-continuous range. The 3-channel procedure increases the duty-cycle for transmission β by a factor of 3 (since 3 beacons are sent every T_a time-units), and this increased duty-cycle leads to essentially the same discovery latency as the original one for typical parameterizations used on smartphones. In BLE, the minimal beacon length is 16 bytes, because of multiple overheads. This further increases β by a factor of around 3 compared to the protocol we described in the previous section.

In summary, compared to an energy-optimal protocol, BLE introduces an overhead of a factor of approximately 6 to the active time spent for transmission. An additional overhead to γ is caused by the random delay ρ, since d_s needs to be increased for guaranteeing the same worst-case latency. On the other hand, some of these overheads improve the reliability of the discovery procedure. In particular, cyclic redundancy check (CRC) allows detecting beacon collisions and the random delay decorrelates the collision probabilities of multiple subsequent beacons from each other. Due to fixed energy overheads that do not depend on the number of transmitted bytes, the overhead in terms of energy is lower than in terms of transmission/reception time.

3.1.2 Android and iOS. While BLE allows essentially any configuration for the tuple $(T_{a,0}, T_s, d_s)$, Android constrains the design space to 3 different settings that determine (T_s, d_s) and another 3 that determine $T_{a,0}$. In addition, there is a dedicated parameterization for the contact tracing service offered by Google and Apple, called the *Google Apple Exposure Notification Service* (GAEN) [7]. Furthermore, Android provides a *batch mode*, where multiple discovered devices are reported jointly with a certain delay. This mode offers for 3 additional configurations of (T_s, d_s). But these are not considered in this article due to lack of documentation on them.

$T_{a,0}$		
152.5 ms	211.25 ms	318.75 ms
417.5 ms	546.25 ms	760 ms
852.5 ms	1022.5 ms	1285.0 ms

Table 2: Advertising intervals supported by IOS.

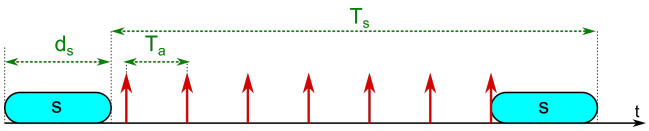


Figure 1: Sequence of beacons (red arrows) with $T_a \leq d_s$ falling into an instance of T_s.

On a smartphone, fixed values for these parameters are not always feasible, because the hardware might be forced to vary them during runtime. First, the Bluetooth system-on-a-chip (SoC) might need to carry out advertising and scanning in parallel to other tasks, e.g., streaming audio to a wireless headphone. As a result, the points in time when both tasks need to be served might overlap. Since the device can carry out only one task at a time, some scheduling is needed to resolve this conflict, which might require adapting the values of T_a, T_s and d_s online. In other words, the effective values of T_a, T_s and d_s that are used could be different from the ones that were chosen by a contact tracing application. In addition, many smartphones, e.g., the Samsung Galaxy S1, share the radio and/or the antenna for realizing IEEE 802.11 (WiFi) and Bluetooth communication [22]. However, it is not possible to e.g., transmit a WiFi frame and a BLE beacon at the same time. Indeed, SoCs that combine BLE and WiFi use arbitration mechanisms for this purpose [21]. To the best of our knowledge, the exact methods used for resolving such conflicts depends on the particular SoC used and hence potentially vary between different smartphone models. In general, there are two possibilities for resolving them. First, the parameters $T_{a,0}$, T_s and d_s could be chosen such that no advertising packet or scan window overlaps with any other task. Second, if this is not possible, an advertising packet could be skipped, sent earlier/later, or the parameters $T_{a,0}$, T_s and d_s might be altered repeatedly on a short-term basis.

The values of T_a, T_s and d_s used in Android smartphones are not specified in the official documentation – the reason for this might lie in the potential need for short-term changes described above. However, Android is an open source software, and information that is not provided in its specification documents can be looked up directly from its source code. The Android source code contains different values of $(T_{a,0}, T_s, d_s)$, which are summarized in Table 1. A pair of values for T_s and d_s can be selected by an application by choosing one of the SCAN_MODE settings, whereas the value of $T_{a,0}$ can be selected by using one of the ADVERTISE_MODE settings.

Since iOS is a closed source software, we cannot obtain its parameter values from the source code. However, Apple recommends certain values of $T_{a,0}$ for gadgets communicating with iOS devices in [2], which are given in Table 2. We assume that iOS devices themselves will also use them. No data is available on T_s and d_s.

Especially on iOS devices, restrictions on using BLE in the background prevent applications from using the original BLE API with the parameter values described above. For example, an early version of the NHS corona app in the UK was reported to work reliably only when the screen is unlocked and the app is visible [23]. To overcome this, the majority of recent apps use GAEN. Here, the operating system carries out the actual contact tracing, while reporting digests to the tracing apps. GAEN uses its own set of parameter values. The advertising interval is specified to lie within 200 to 270 ms and the scan interval to be at most 5 minutes, whereas no information on the scan window is given [7]. Hence, the exact values that are actually used are not fully specified. Further, directly measuring these values is complicated by Google/Apple's policy that grants GAEN access only to applications that are government authorized. However, experimental studies on GAEN have nevertheless been carried out [13, 14]. We have estimated the parameter values given in Table 1 from publicly available logs[1] of received packets. Here, we have assumed that multiple reported receptions in close temporal proximity belong to the same scan window, which allowed us to infer T_a, T_s and d_s based on 119,550 reported receptions. We have excluded scan windows with few (i.e., less than 5) receptions. For every pair of handsets contained in the dataset (except the *Huawei P Smart*, which showed inconsistent results), we have computed the median value of T_a, T_s and d_s. The values given in Table 1 are the mean values of those medians. The parameter values can vary significantly over time. It is worth mentioning that when some other app running on the phone activates BLE scanning in a certain mode, GAEN will re-use the selected parameter values and therefore deviate from the values given in Table 1.

3.2 Performance Evaluation

In this section, we evaluate how BLE ND performs on smartphones. Towards this, we assume 1) a packet length of 16 byte and 2) the values from Table 1 for Android and the values of $T_{a,0}$ from Table 2 for iOS.

3.2.1 Discovery Latency. We now study the discovery latency for two devices. Most of the $T_{a,0}$-values for Android and iOS fulfill $T_a < d_s$. This is illustrated in Figure 1. Once two devices are within their reception range, the first advertising beacon of one device falls within an arbitrary instance of a scan interval. Since $T_a < d_s$, the latency measured from the first beacon to the successfully received one is limited by roughly $T_s - d_s$ time units. Because both devices might have been brought into range by up to $T_{a,0} + 10$ ms before the first beacon was sent, the worst-case latency is bounded by approximately $L = T_s - d_s + T_{a,0} + 10$ ms (cf. Figure 1). As an example, for the SCAN_MODE_LOW_POWER setting and a value of $T_{a,0} = 100$ ms, the worst-case latency is approximately 4,718ms.

On the other hand, for some other settings, such as ADVERTISE_MODE_LOW_POWER in combination with SCAN_MODE_LOW_POWER, $T_a > d_s$. Here, the first scan window might be missed in some cases, and only a later scan window can lead to a successful discovery. Figure 2a depicts the simulated discovery latencies measured from the time the first beacon is sent, after which two devices come within range. This is with the SCAN_MODE_LOW_POWER setting. Similarly, Figure 2b depicts the discovery latencies for the SCAN_MODE_BALENCED setting. For every value of $T_{a,0}$ supported by BLE in the depicted range, we have carried out 1,000 simulations. For each of them, we have computed the maximum and mean latency. The solid darker red vertical lines correspond to the values of $T_{a,0}$ supported by the native BLE interface on Android, and the dashed lines in light red are those supported by iOS.

[1]available via https://github.com/sftcd/gaen-pairwise-1m

Scan Mode	nRF52832	BLE112
LOW_POWER	0.52 – 0.57 %	2.13 – 2.35 %
BALANCED	1.30 – 1.35 %	5.30 – 5.52 %
LOW_LATENCY	5.20 – 5.25 %	21.14 – 21.36 %
GAEN	0.09 %	0.38 %

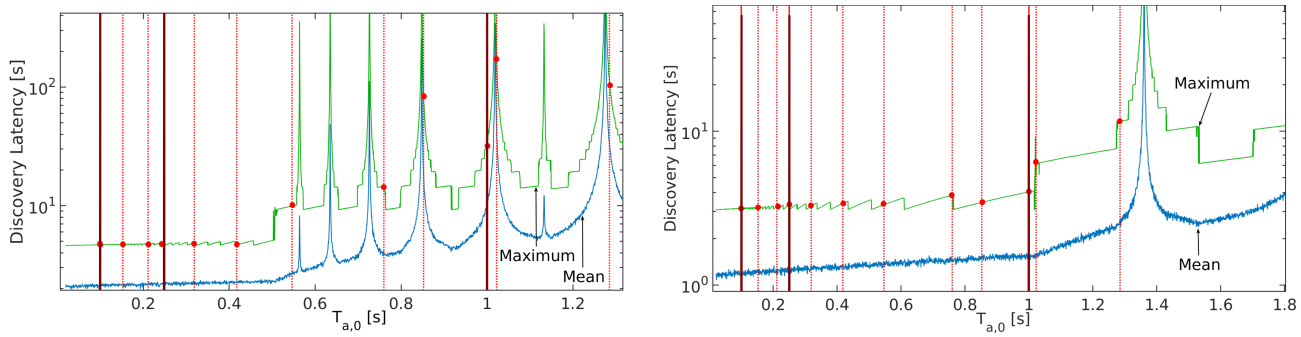
Table 3: Reduction of the battery runtime by continuous contact tracing using different Android scan modes.

Let us first consider the SCAN_MODE_LOW_POWER setting. As can be seen, for values with $T_a < d_s$, the maximum latency is approximately 5 s. However, for $T_a > d_s$, some parameterizations lead to high maximum and mean latencies. For example, for $T_{a,0} =$ 1,022.5ms, which is supported by Android, the maximum resulting latency is 172.5 s. For the purpose of contact tracing, such a latency could be unacceptable, since a close contact of less than 3 minutes could already be highly relevant for a virus transmission. Recall from our previous discussion that because of issues such as resource sharing, smartphones might also deviate from the given parameter values, and hence there are no guarantees that these maximum latencies are never exceeded by any smartphone model.

For the SCAN_MODE_BALANCED setting, the overall situation looks similar (cf. Figure 2b), but a larger fraction of the values of $T_{a,0}$ lead to latencies below 5 s.

Different smartphone manufacturers might have altered these values in their adopted versions of Android, or might do this in the future. Therefore, compatibility among all different smartphone models – for the purpose of estimating a worst-case discovery latency – is also not guaranteed. Given of all this, we have to consider a smartphone as a device that maintains a certain maximum discovery latency in *most* cases, while no absolute guarantees can be given for the worst case.

For GAEN, the worst-case latency lies around 5 min. This is because it schedules one scan window that is significantly larger than the distance between two consecutive advertising packets every at most 5 min in the worst case. We found that T_a, T_s and d_s vary significantly over time and also found instances where $T_a > d_s$ in the dataset. Hence, a latency of more than 5 min might also be possible in rare cases. However, we could not verify from the dataset whether two consecutive packets were indeed scheduled such that their time difference exceed the shortest measured instance of d_s. Other effects might also be possible – such as packets colliding with those of other devices, or devices being intermittently out of range, leading to failed receptions. The latency bound of 5 min also limits the granularity of the contact duration estimation to up to 5 min, which could lead to missing multiple brief contacts.



(a) SCAN_MODE_LOW_POWER **(b) SCAN_MODE_BALANCED**

Figure 2: Simulation of maximum and mean discovery latencies in Android for different values of $T_{a,0}$. The vertical lines depict values of $T_{a,0}$ used by Android (solid lines) and iOS (dashed lines).

3.2.2 Energy Consumption. We now study how the settings from Table 1 impact the energy consumption in the case of Android phones. We have computed the mean current draw I_{BLE} of a BLE radio that scans and advertises according to the different settings. The energy needed for the phone's CPU to wake up and gather the data from the radio is not included.

Different smartphones use different radios with different firmware and, accordingly, have different energy consumption. We computed the energy consumption for two SoCs with published energy data. But we did not see any smartphone using these chips. Smartphones typically use dual-mode chips that support WiFi and Bluetooth, for which detailed energy information is not widely available. Therefore, we study two different, well-chosen Bluetooth radios, and assume that the energy consumption of the SoCs used on smartphones lie in between. First, we consider a Bluegiga BLE112 module, which was among the first BLE radios available. In addition, we consider the more recent Nordic nRF52832 SoC.

Estimation for the BLE112 device was been carried out using the energy model proposed in [12]. For the nRF52832 SoC, we have used the energy model provided by the device manufacturer for advertising [19], which we have combined with values from the device's datasheet [18].

We have put I_{BLE} in relation to the mean current draw of the smartphone. For this purpose, we have assumed that the smartphone is equipped with a 3000 mAh battery. We further assumed that this capacity is drained within 24 h when contact tracing is not carried out, which leads to an average current consumption of $I_P = 3000\,\text{mAh}/24\,\text{h}$ of the smartphone. We can now form the quotient I_{BLE}/I_P, which gives us the percentage of time within which continuous contact tracing will drain the battery earlier.

Table 3 shows the results of this computation. The range of values shown for each scan setting comprises all different advertising intervals $T_{a,0}$ supported by Android, iOS and GAEN. For the BLE112 radio, the SCAN_MODE_LOW_LATENCY setting reduces the battery lifetime by about 20 %, which will be noticeable in practice. The SCAN_MODE_BALANCED setting reduces the battery lifetime by about 5 %, and the SCAN_MODE_LOW_POWER by about 3 %. GAEN drains the battery by less than 0.4 % earlier, which can not be noticed by an user. For the nRF51822 radio, the energy consumption is approximately 75 % lower in all modes of operation.

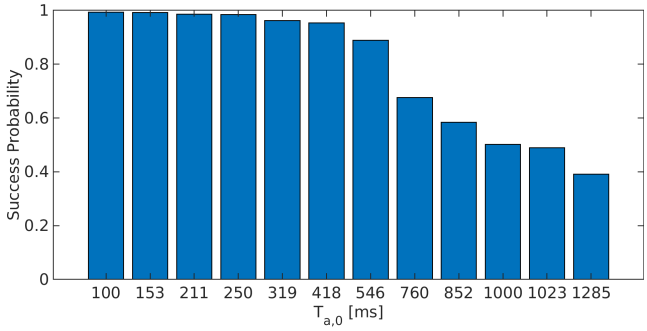


Figure 3: Probabiltiy of discovery in a crowd of 100 subjects for SCAN_MODE_LOW_POWER.

It should be mentioned that compared to a ND solution with optimal energy efficiency (cf. [11]), the parameterizations supported by Android, iOS and GAEN require significantly more energy for providing the same worst-case latency. However, the overall energy demand of the Bluetooth radio is small compared to the capacity of batteries in most currently available smartphones.

3.2.3 Collision Behavior. If two or more devices transmit a beacon at the same time, these beacons will overlap and hence collide. With high probability, these beacons will then not be received successfully, even when they coincide with a scan window. This might increase the discovery latency, or even prevent discovery in some cases. The choice of $T_{a,0}$ impacts the collision rate, since it affects the fraction of time β during which the channel is busy. To quantify this, we have simulated 10,000 discovery procedures for each parameterization. To cover the absolute worst case, here we assumed that up to 100 devices can interfere with each other. We have considered a discovery procedure as successful, if at least one non-colliding beacon coincided with a scan window within 10 s. This simulation can be regarded as a worst-case estimation. In particular, our simulation considers every pair of simultaneously sent packets by different devices as a collision that leads to a loss of both packets. In practical setups, one of these two packets might still be received successfully, e.g., because the signal of the interfering packet has a lower power than the successful one. The purpose of this study is therefore to show that the resulting performance is sufficient for practical needs, even under pessimistic assumptions.

For the SCAN_MODE_LOW_LATENCY and SCAN_MODE_BALANCED settings, assuming that every smartphone uses the same settings, the resulting success probability is 100 % for all values of $T_{a,0}$ supported by Android and iOS, barring one exception for SCAN_MODE_BALANCED with $T_{a,0} = 1.285$ s, where the success probability is reduced to 96.4 %. Figure 3 depicts the probabilities for the SCAN_MODE_LOW_POWER setting. As can be seen, the success probabilities vary between 99.4 % and 39.1 %. For values of $T_{a,0} > 512$ ms, the low success probability is caused both by colliding beacons and by the theoretical worst-case latency (i.e., the latency when no collision occur) exceeding 10 s. For GAEN, when assuming the interval lengths given by Table 1, the probability that at least one beacon is received in every scan window is also essentially 100 %. The reasons for the high success probability for most parameterizations are a) the relatively low channel utilization $\beta = \omega/T_a$ of each radio, and b) the large scan window d_s in combination with the random delay. Since multiple beacons are sent within every scan window, the probability that one of them is received without colliding with a beacon from a different device is high. Since the channel is only moderately used in GAEN and the other available settings, ubiquitous tracing devices are not expected to disturb any other BLE applications, neither on the same device and nor on devices in the surrounding environment.

3.3 Other Limitations

Besides the limitations concerning discovery latency, the following additional limitations reduce the performance and reliability of contact tracing on smartphones.

(1) For estimating the proximity between two subjects, every received packet comes with a certain received signal strength indicator (RSSI), which represents the *path loss* between a sender and the receiver. Proximity estimation is based on the assumption that every RSSI corresponds to a unique distance. However, the RSSI is also influenced by multiple other factors, e.g., the smartphone model, the orientation of the sending and receiving antennas, the environment (e.g., metal parts in vicinity) and the wireless channel used. While some of these uncertainties can be mitigated through various techniques, the following problem will always remain.
Whenever the wireless signal has to pass through the human body, the attenuation is much larger than in free space. For example, [1] reports an attenuation of up to 19.2 dB between the chest and the back of a human body. How does this impact distance estimation in contact tracing? We placed two identical smartphones on a table at a distance of around 10 cm. When running the ITO demonstration app for contact tracing [9], our experiments showed that the estimated distance was around 50 cm when both smartphones were within line of sight. But when a human arm was placed between them, the estimated distance increased to almost 5 m. Hence, in some situations, a smartphone would classify a contact as being "far away", while it actually being close. For example, consider two people walking side-by side and holding their hands, but having their smartphones in opposite pockets. Though this would represent a relevant contact, the obstructed line of sight would, with high probability, lead to a large distance estimation and hence to a misclassification of the contact. Indeed, experimental studies using GAEN have shown that distance estimation is prone to errors. For example, [13] reports that based on data obtained in an experiment involving multiple smartphone users located in a light-rail tram, the German tracing app would trigger no exposure notification at all, while the Italian and Swiss apps would lead to 50 % true and 50 % false positives. These results suggest that when many metal parts are in the surrounding environment, distance estimation using smartphones is not reliable.

(2) For tracing all relevant contacts using a wireless solution, it is important that as many people as possible participate. However, for contact tracing on smartphones, an application (app) needs to be installed, the appropriate permissions need to be granted, and the app needs to be activated. Hence, a certain part of the population may be physically or mentally unable to handle a smartphone, e.g., small children or the elderly.

3.4 Summary

Today, contact tracing apps are heavily reliant on GAEN and can hence reliably detect only those contacts whose durations are at least $\approx$ 5 min long. Therefore, brief contacts, e.g., in the course of a short haul in a crowded subway, might remain unnoticed in spite of a potentially significant transmission risk. Energy consumption is only of concern in the SCAN_MODE_LOW_LATENCY setting.

Overall, our performance evaluation shows that the SCAN_MODE_BALANCED-setting guarantees low latencies of around 5 s, if advertising intervals below 1 s are used. At the same time, all such configurations would provide a success probability of essentially 100 % when multiple phones are within range. Also, the energy consumption would remain at a reasonable level. Therefore, contact tracing applications on smartphones would benefit significantly from Apple and Google providing access to the SCAN_MODE_BALANCED setting and to at least one of the supported advertising intervals below 1 s. Providing an optimized and dedicated parametrization for contact tracing would further decrease the discovery latencies. For example, a shorter scan window could be scheduled more frequently, while mostly preserving energy. This could lead to worst-case latencies of below 5 s, if the advertising interval would also be shortened accordingly. Further, the accuracy of distance estimation could potentially be improved by gathering a larger number of RSSI samples. This could be achieved by allowing a tracing app to temporarily listen continuously, once an initial contact has been established.

Both theoretical considerations and practical experiments suggest that at least in certain situations, e.g., in a tram or in a bus, smartphone-based distance estimation cannot always distinguish between relevant and irrelevant contacts. Overall, smartphone-based contact tracing is a *best-effort* mechanism without any guarantees. But if a dedicated device for tracing is designed, these limitations could be mitigated effectively, as described next.

4 CONTACT TRACING USING WEARABLES

In this section, we discuss a wearable solution and outline how it could mitigate the problems described in the previous section. Unlike most existing solutions, e.g., [17], we propose a wearable that uses a custom-designed wireless protocol and does not rely on Bluetooth. Consider a wrist-worn bracelet, a necklace or a sticker.

It would consist of low-cost hardware, e.g., a wireless radio, an accelerometer and a battery. Such a wearable solution would mitigate the main problems associated with smartphone-based solutions. First, it could use the entire design space of ND protocols and therefore provide worst-case latencies of around 2 s. Second, **distance estimation** also need not rely on Bluetooth. In particular, Ultra-Wideband (UWB) in combination with Time-of-Flight could be used for distance estimation. Ranging experiments show that such methods achieve reasonable accuracy even when the human body attenuates the signal [24]. Recent studies indicate that they can also improve accuracy in contact tracing scenarios [5]. Also, the combination of Bluetooth and UWB for contact tracing is currently being studied [8]. Further, frequency bands other than the 2.4 GHz used by Bluetooth can be used, which are significantly less susceptible to in-body attenuation and multipath propagation. A wearable is active whenever worn and hence **does not require any handling**. Thus, faulty handling is ruled out. It could also detect certain events of particular relevance for transmission, e.g., two subjects touching each other. In addition, the **battery lifetime** of such a wearable could be drastically increased compared to a smartphone. For example, consider a low-cost nRF52832 radio. We can estimate its energy consumption based on the data provided from the device manufacturer [19] and some additional overheads. We assume a 380 mAh battery, which is often found in smartwatches. For guaranteeing a discovery latency of 2 s, using the parameterizations from [10], a battery runtime of up to 235 days can be achieved, which could be further extended by switching the device off automatically when not being worn.

In summary, a dedicated wearable device could be designed to optimize every aspect of the tracing procedure and therefore potentially provide a significantly higher tracing reliability. Nevertheless, there are also multiple caveats that are frequently discussed in the context of wearables. First, wearables need to be mass-produced and deployed, while smartphones are already prevalent. However, wearables for other purposes are already being mass-produced, e.g., in the form of watches, earphones or spectacles. Hence, the necessary facilities for scaling up the production of tracing wearables are already present. Contact tracing functionality can also be integrated into wearables for different purposes (e.g., fitness tracking), which might increase the prevalence of tracing devices. Second, privacy concerns often prevent the deployment of tracing apps. Here, wearables are advantageous because they are dedicated low-content devices, which do not store any unnecessary private data, such as pictures or personal messages. Nevertheless, users might still mistrust a dedicated body-worn device, because it might potentially also collect data not needed for contact tracing.

5 CONCLUDING REMARKS

We have studied the technical feasibility of using smartphones for contact tracing. Even though their radios support the essential features necessary for contact tracing, many contacts potentially cannot be detected due to the high worst-case latency, which lies around 5 min for tracing solutions using GAEN. While this could be mitigated if tracing apps could make full use of the existing BLE APIs, contact classification using distance estimation cannot be done with sufficient accuracy in all situations. A dedicated wearable that does not rely on Bluetooth could overcome this limitation and increase the effectiveness of electronic contact tracing.

REFERENCES

[1] A. Alomainy, Y. Hao, A. Owadally, C. G. Parini, Y. Nechayev, C. C. Constantinou, and P. S. Hall. 2007. Statistical analysis and performance evaluation for on-body radio propagation with microstrip patch antennas. *IEEE Transactions on Antennas and Propagation* 55, 1 (2007), 245–248.

[2] Apple, Inc. 2019. Accessory Design Guidelines for Apple Devices. available via https://developer.apple.com/accessories/Accessory-Design-Guidelines.pdf.

[3] P. Doung-ngern, R. Suphanchaimat, A. Panjangampatthana, C. Janekrongtham, D. Ruampoom, N. Daochaeng, and N. Eungkanit et al. 2020. Case-control study of use of personal protective measures and risk for SARS-CoV 2 infection, Thailand. *Emerging Infectious Diseases* 26, 11 (2020), 2607.

[4] P. Dutta and D. Culler. 2008. Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*. 71–84.

[5] B. Etzlinger, B. Nußbaummüller, P. Peterseil, and K. A. Hummel. 2021. Distance Estimation for BLE-based Contact Tracing – A Measurement Study. *arXiv.org 2101.09075* (2021).

[6] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser. 2020. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science* (2020).

[7] Google. 2020. Contact Tracing Bluetooth Specification. available via https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf.

[8] T. Istomin, E. Leoni, D. Molteni, A.L. Murphy, G.P. Picco, and M. Griva. 2021. Janus: Efficient and Accurate Dual-radio Social Contact Detection.

[9] ITO Consortium. 2020. ITO demonstration app. https://www.ito-app.org.

[10] P.H. Kindt, S. Narayanaswamy, M. Saur, and S. Chakraborty. 2020. Optimizing BLE-Like Neighbor Discovery. *IEEE Transactions on Mobile Computing, to appear* (2020).

[11] P. H. Kindt and S. Chakraborty. 2019. On Optimal Neighbor Discovery. In *ACM Special Interest Group on Data Communication (SIGCOMM)*.

[12] P. H. Kindt, D. Yunge, R. Diemer, and S. Chakraborty. 2020. Energy Modeling for the Bluetooth Low Energy Protocol. *ACM Trans. Embed. Comput. Syst. (TECS)* 19, 2 (March 2020).

[13] D. J: Leith and S. Farrell. 2020. Measurement-based evaluation of Google/Apple Exposure Notification API for proximity detection in a light-rail tram. *PloS one* 15, 9 (2020).

[14] D. J. Leith and S. Farrell. 2021. GAEN Due Diligence: Verifying The Google/Apple Covid Exposure Notification API. *Network and Distributed System Security Symposium (NDSS)* (2021).

[15] D. Lu. 2021. Covid Delta variant is "in the air you breathe": what you need to know about Sydney outbreak strain. https://bit.ly/3xoaFZ0.

[16] Bluetooth SIG. 2019. Specification of the Bluetooth System 5.2. Volume 0, available via bluetooth.org.

[17] MIT Media Lab. 2020. Open Badges Project. https://www.media.mit.edu/projects/open-badges.

[18] Nordic Semiconductor. 2016. nRF52832 Product Specification v1.1. available via https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.1.pdf.

[19] Nordic Semiconductor. 2020. Online Power Profiler. available via https://devzone.nordicsemi.com/nordic/power.

[20] D. Oberhagemann. 2012. Static and dynamic crowd densities at major public events. *Vereinigung zur Förderung des Deutschen Brandschutzes e. V. (vfdb), German Fire Protection Association, Technical-scientific advisory board (TWB)* 13 (2012).

[21] Silicon Laboratories Inc. 2020. AN1128: Bluetooth Coexistence with Wi-Fi. https://www.silabs.com/documents/public/application-notes/an1128-bluetooth-coexistence-with-wifi.pdf.

[22] TechInsights Inc. 2020. Samsung Galaxy S10+ Teardown. available via https://www.techinsights.com/blog/samsung-galaxy-s10-teardown.

[23] The Verge. 2020. Without Apple and Google, the UK's contact-tracing app is in trouble. www.theverge.com/2020/5/5/21248288.

[24] Q. Tian, I. Kevin, K. Wang, and Z. Salcic. 2018. Human body shadowing effect on UWB-based ranging system for pedestrian tracking. *IEEE Transactions on Instrumentation and Measurement* 68, 10 (2018), 4028–4037.

[25] C. Wymant, L. Ferretti, D. Tsallis, M. Charalambides, L. Abeler-Dörner, D. Bonsall, R. Hinch, M. Kendall, L. Milsom, M. Ayres, et al. 2021. The epidemiological impact of the NHS COVID-19 App. *Nature* (2021), 1–8.